RAMEEZ UR REHMAN SIDDIQUI DS-041

ASSIGNMENT#1

"TRAIN AN ARTIFICIAL NEURON ON THE PRINCIPLE OF TWO INPUT AND/OR GATE USING ERROR CORRECTION LEARNING"

PERCEPTRON MODEL :

```
import pandas as pd
import numpy as np

class Perceptron:
    def __init__(self, eta: float=None, epochs: int=None):

# small randomly initiallized weights
        self.weights = np.random.randn(3) * 1e-4
        training = (eta is not None) and (epochs is not None)
        if training:
            print(f"initial weights before training: \n{self.weights}\n")
        self.eta = eta
        self.epochs = epochs

    def _z_outcome(self, inputs, weights):
        return np.dot(inputs, weights)

# hard limit activation function y = {0 z>=0 --> 1 and z<0 --> 0
    def hard_limiter(self, z):
        return np.where(z >= 0, 1, 0)

    def fit(self, X, y):
        self.X = X
        self.y = y

        X_with_bias = np.c_[self.X, -np.ones((len(self.X), 1))]
        print(f"X with bias: \n{X_with_bias}")

        for epoch in range(self.epochs):
            print("--"*10)
            print(f"for epoch >> {epoch}")
            print("--"*10)

            z = self._z_outcome(X_with_bias, self.weights)
            y_hat = self.hard_limiter(z)
            print(f"predicted value after forward pass: \n{y_hat}")
```

```
#Error Equation --> error = desired - actual
            self.error = self.y - y_hat
            print(f"error: \n{self.error}")

#Weight Adjustment Equation --> delta rule
            self.weights = self.weights + self.eta * np.dot(X_with_bias.T, 
self.error)
            print(f"updated weights after epoch: {epoch + 1}/{self.epochs}: 
\n{self.weights}")
            print("##"*10)

    def predict(self, X):
        X_with_bias = np.c_[X, -np.ones((len(X), 1))]
        z = self._z_outcome(X_with_bias, self.weights)
        return self.hard_limiter(z)
    def total_loss(self):
        total_loss = np.sum(self.error)
        print(f"\ntotal loss: {total_loss}\n")
        return total_loss

def prepare_data(df, target_col="y"):
    X = df.drop(target_col, axis=1)

    y = df[target_col]

    return X, y
```

AND GATE :

```
AND = {
    "x1": [0,0,1,1],
    "x2": [0,1,0,1],
    "y" : [0,0,0,1]
}

df_AND = pd.DataFrame(AND)

df_AND

X, y = prepare_data(df_AND)

ETA = 0.1 # 0 and 1
```

```python
EPOCHS = 4

model_and = Perceptron(eta=ETA, epochs=EPOCHS)
model_and.fit(X, y)

_ = model_and.total_loss()

model_and.save(filename="and.model")
reload_model_and = Perceptron().load(filepath="model/and.model")
reload_model_and.predict(X=[[1,1]])
```

OUTPUTS :

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

(masters_env) PS D:\NED masters\my masters\assignment1> & "d:/NED masters/my masters/assignment1/m
masters/assignment1/final_and.py"
initial weights before training:
[-3.28353491e-05 -1.39774453e-04  1.67409495e-04]

X with bias:
[[ 0.  0. -1.]
 [ 0.  1. -1.]
 [ 1.  0. -1.]
 [ 1.  1. -1.]]
--------------------
for epoch >> 0
--------------------
predicted value after forward pass:
[0 0 0 0]
error:
0    0
1    0
2    0
3    1
Name: y, dtype: int64
updated weights after epoch: 1/4:
[ 0.09996716  0.09986023 -0.09983259]
####################
--------------------
for epoch >> 1
--------------------
predicted value after forward pass:
[1 1 1 1]
error:
0   -1
1   -1
2   -1
3    0
Name: y, dtype: int64
updated weights after epoch: 2/4:
[-3.28353491e-05 -1.39774453e-04  2.00167409e-01]
venv)  ⊗ 0 ⚠ 0
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

3    0
Name: y, dtype: int64
updated weights after epoch: 2/4:
[-3.28353491e-05 -1.39774453e-04  2.00167409e-01]
####################
--------------------
for epoch >> 2
--------------------
predicted value after forward pass:
[0 0 0 0]
error:
0    0
1    0
2    0
3    1
Name: y, dtype: int64
updated weights after epoch: 3/4:
[0.09996716 0.09986023 0.10016741]
####################
--------------------
for epoch >> 3
--------------------
predicted value after forward pass:
[0 0 0 1]
error:
0    0
1    0
2    0
3    0
Name: y, dtype: int64
updated weights after epoch: 4/4:
[0.09996716 0.09986023 0.10016741]
####################

total loss: 0

(masters_env) PS D:\NED masters\my masters\assignment1>
```

OR GATE :

```
OR = {
    "x1": [0,0,1,1],
```

```
    "x2": [0,1,0,1],
    "y" : [0,1,1,1]
}

df_OR = pd.DataFrame(OR)

df_OR

X, y = prepare_data(df_OR)

ETA = 0.1 # 0 and 1
EPOCHS = 6

model_or = Perceptron(eta=ETA, epochs=EPOCHS)
model_or.fit(X, y)

_ = model_or.total_loss()

model_or.save(filename="or.model", model_dir="model_or")
```

OUTPUT:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
(masters_env) PS D:\NED masters\my masters\assignment1> & "d:/NED masters/my masters/assignment1/ma
masters/assignment1/final_or.py"
initial weights before training:
[7.39059772e-05 1.28111270e-04 1.28476124e-04]

X with bias:
[[ 0.  0. -1.]
 [ 0.  1. -1.]
 [ 1.  0. -1.]
 [ 1.  1. -1.]]
--------------------
for epoch >> 0
--------------------
predicted value after forward pass:
[0 0 0 1]
error:
0    0
1    1
2    1
3    0
Name: y, dtype: int64
updated weights after epoch: 1/6:
[ 0.10007391  0.10012811 -0.19987152]
####################
--------------------
for epoch >> 1
--------------------
predicted value after forward pass:
[1 1 1 1]
error:
0   -1
1    0
2    0
3    0
Name: y, dtype: int64
updated weights after epoch: 2/6:
[ 0.10007391  0.10012811 -0.09987152]
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

3    0
Name: y, dtype: int64
updated weights after epoch: 4/6:
[0.10007391 0.10012811 0.00012848]
####################
--------------------
for epoch >> 4
--------------------
predicted value after forward pass:
[0 1 1 1]
error:
0    0
1    0
2    0
3    0
Name: y, dtype: int64
updated weights after epoch: 5/6:
[0.10007391 0.10012811 0.00012848]
####################
--------------------
for epoch >> 5
--------------------
predicted value after forward pass:
[0 1 1 1]
error:
0    0
1    0
2    0
3    0
Name: y, dtype: int64
updated weights after epoch: 6/6:
[0.10007391 0.10012811 0.00012848]
####################

total loss: 0

(masters env) PS D:\NED masters\my masters\assignment1>
```

MY COMMENTS :

So the training of artificial neuron on the principle of 2 input AND gate using ECL is done above, The coding is done in python Language.

We can see clearly, after epoch 4 in "AND GATE" we got "zero error" so training would be stopped at this point and we got:

Y = $(f)\begin{bmatrix}0.099 & 0.098 & 0.1\end{bmatrix}\begin{bmatrix}x0\\x1\\x2\end{bmatrix}$

And in "OR GATE" we got "zero error" after 6 epochs so training would be stopped at this point and we got:

Y = $(f)\begin{bmatrix}0.1 & 0.1 & 0.0001\end{bmatrix}\begin{bmatrix}x0\\x1\\x2\end{bmatrix}$

Hard limit activation function is used in it.

when activity is less than zero the output would be zero whereas output would be one for activity greater or equal to zero.

Small randomly initialized weights are used with bias included and learning rate (eta) is taken as 0.1.